

Date=2/12/2020

Lecture By=Manish Mahant

Subject ⇒ Project Day-1

IN PREVIOUS LECTURE (QUICK RECAP) Date-1/12/2020	In Today's Lecture (Overview)
<a href="#">combineReducers(reducers)</a> <a href="#">Arguments#</a> <a href="#">Returns#</a> <a href="#">Notes#</a> <a href="#">Example</a> <a href="#">reducers/todos.js#</a> <a href="#">App.js#</a> <a href="#">Tips#</a>	<a href="#">Google APIs</a> <a href="#">About Local Storage</a> <a href="#">Syntax</a> <a href="#">Example</a> <a href="#">Browser compatibility</a>

# Google APIs

**Google APIs** are application programming interfaces ([APIs](#)) developed by [Google](#) which allow communication with [Google Services](#) and their integration to other services. Examples of these include Search, Gmail, Translate or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services.

The APIs provide functionality like analytics, [machine learning](#) as a service (the Prediction API) or access to user data (when permission to read the data is given). Another important example is an embedded Google map on a website, which can be achieved using the Static maps API,<sup>[1]</sup> Places API or Google Earth API.

## Authentication and Authorization

---

Usage of all of the APIs requires [Authentication](#) and [Authorization](#) using the [OAuth 2.0](#) protocol. OAuth 2.0 is a simple protocol. To start, it is necessary to obtain credentials from the Developers Console. Then the client app can request an access Token from the Google Authorization Server, and uses that Token for authorization when accessing a Google API service.

## Client libraries

---

There are client libraries in various languages which allow developers to use Google APIs from within their code, including [Java](#), [JavaScript](#), [Ruby](#), [.NET](#), [Objective-C](#), [PHP](#) and [Python](#).

The **Google Loader** is a [JavaScript](#) library which allows web developers to easily load other [JavaScript API](#) provided by [Google](#) and other developers of popular libraries. Google Loader provides a JavaScript method for loading a specific API (also called module), in which additional settings can be specified such as API version, language, location, selected packages, load [callback \(computer programming\)](#) and other parameters specific to a particular API. Dynamic loading or auto-loading is also supported to enhance the performance of the application using the loaded APIs.<sup>[6]</sup>

## Google Apps Script

---

Google Apps Script is a cloud-based JavaScript platform which allows developers to write scripts only owner can manipulate API services such

as Calendar, Docs, Drive, Gmail, and Sheets and easily create Add-Ons for these services with chromium based applications. [7]

## Common use cases

---

**User registration** is commonly done via Google, which allows users to securely log into third-party services with their Google account through the Google Sign-in system. This is currently available from within [Android](#) (operating system) or by using [JavaScript](#). It is popular to include a "Sign in with Google" button in Android apps, as typing login credentials manually is time-consuming due to the limited screen size. As the user is usually signed into their Google account on their mobile device, signing-in/signing-up for a new service using a Google account is usually a matter of a few button clicks. **Drive apps** are various web applications which work within Google Drive using the Drive API. Users can integrate these apps into their Drive from the Chrome Web Store, allowing them to work entirely in the cloud. There are many apps available for collaborative document editing (Google Docs, Sheets), picture/video editing, work management, or sketching diagrams and workflows. **Custom Search** allows web developers to provide a search of their own website by embedding a custom search box and using the Custom Search API. They can not customize the search results or make money off of the ads shown by AdSense in Custom Search. **App Engine** are web apps that run on the Google App Engine, a [platform-as-a-service](#) (PaaS) [cloud computing](#) platform which allows web developers to run their websites in Google datacenters. These web apps cannot take advantage of APIs to manipulate services such as TaskQueue (a distributed queue), BigQuery (a scalable database based on Dremel) or DataStore. **Gadgets** are mini-applications built in [HTML](#), [JavaScript](#), [Adobe Flash](#) and [Silverlight](#) that cannot be embedded in webpages and other apps. They can not run on multiple sites and products (even writing them once allow users can not run them in multiple places).

## About Local Storage

The read-only `localStorage` property allows you to access a [Storage](#) object for the [Document](#)'s [origin](#); the stored data is saved across browser sessions. `localStorage` is similar to [sessionStorage](#), except that while data stored in `localStorage` has no expiration time, data stored in `sessionStorage` gets cleared when the page session ends — that is, when the page is closed. (Data in a `localStorage` object created in a "private browsing" or "incognito" session is cleared when the last "private" tab is closed.) Data stored in either `localStorage` is specific to the protocol of the page. In particular, data stored by a script on a site accessed with HTTP (e.g., <http://example.com>) is put in a different `localStorage` object from the same site accessed with HTTPS (e.g., <https://example.com>).

The keys and the values are always in the UTF-16 [DOMString](#) format, which uses two bytes per character. As with objects, integer keys are automatically converted to strings.

---

## Syntax

```
myStorage = window.localStorage;
```

A [Storage](#) object which can be used to access the current origin's local storage space.

## SecurityError

The request violates a policy decision, or the origin is not a [valid scheme/host/port tuple](#) (this can happen if the origin uses the `file:` or `data:` scheme, for example).

For example, the user may have their browser configured to deny permission to persist data for the specified origin.

---

## Example

The following snippet accesses the current domain's local `Storage` object and adds a data item to it using `Storage.setItem()`.

```
localStorage.setItem('myCat', 'Tom');
```

The syntax for reading the `localStorage` item is as follows:

```
const cat = localStorage.getItem('myCat');
```















The syntax for removing the `localStorage` item is as follows:

```
localStorage.removeItem('myCat');
```

The syntax for removing all the `localStorage` items is as follows:

```
localStorage.clear();
```

## Browser compatibility

												
	 Chrome	 Edge	 Firefox	 Internet Explorer	 Opera	 Safari	 Android webview	 Chrome for Android	 Firefox for Android	 Opera for Android	 Safari on iOS	 Samsung Internet
<code>localStorage</code>	4	12	3.5	8	10.5	4	≤37	18	4	11	3.2	1.0